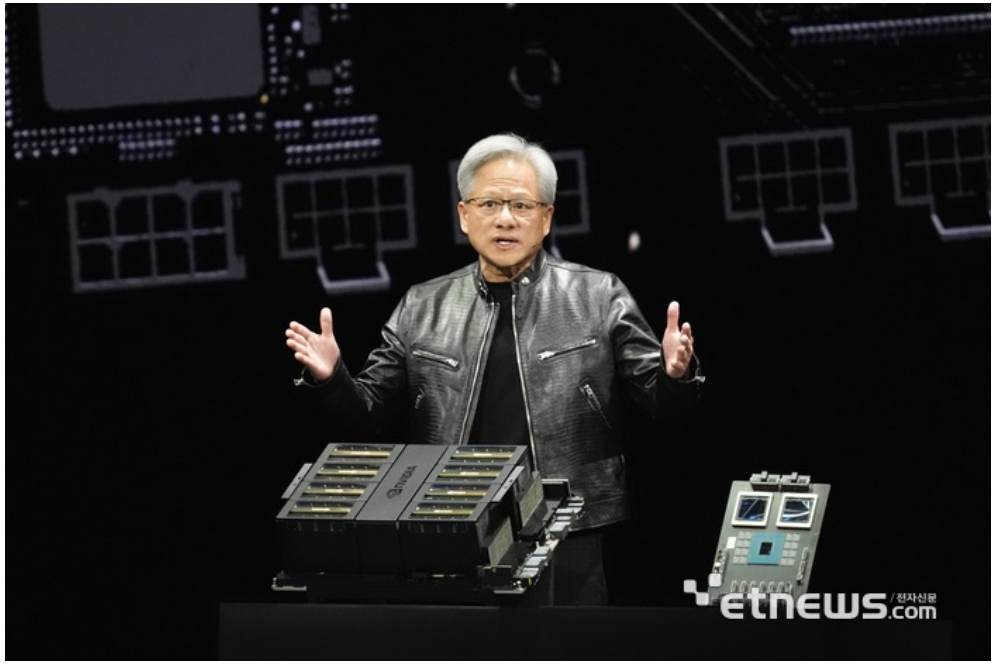
GDSC Seminar:
GPU-efficient ML

고려대학교 정보대학
Korea University
College of Informatics

전병우
ipcs@korea.ac.kr

Why GPU resource is important?

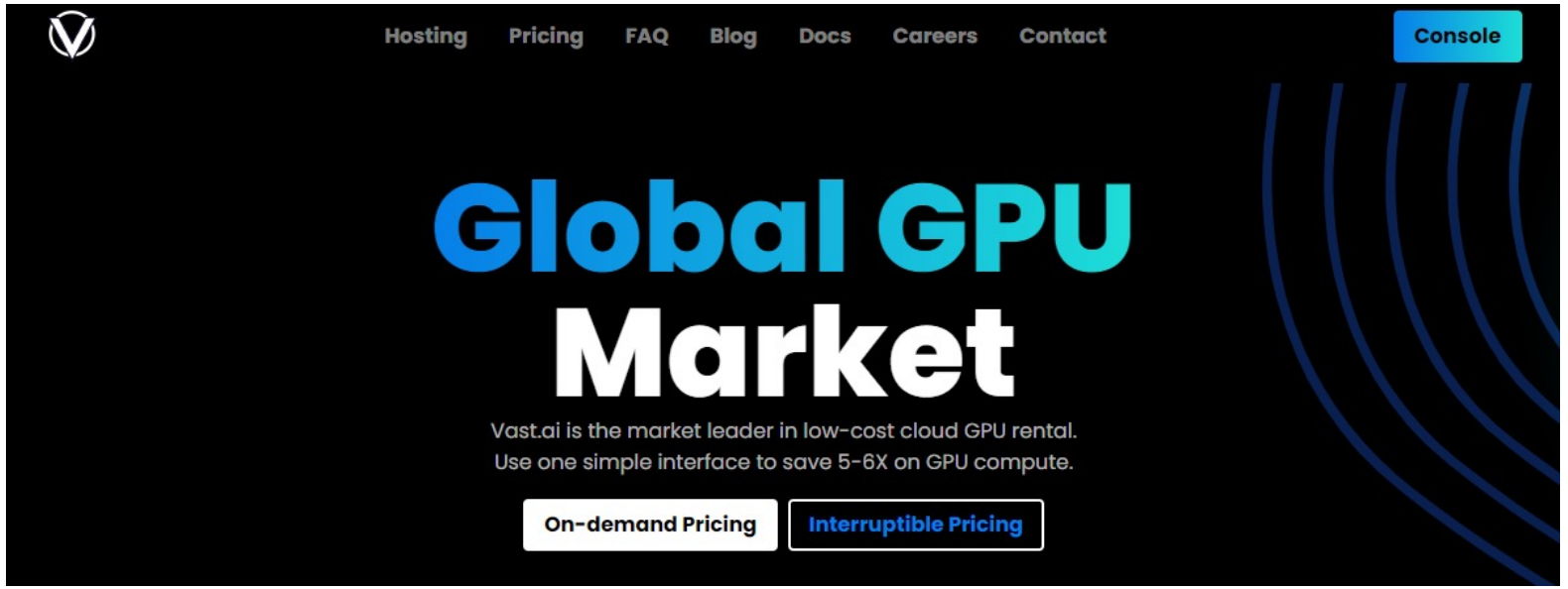
- 지난 10년간의 NVIDIA GPU의 성장은 deep learning revolution을 주도했다.
- GPU의 VRAM이 커지면서 GPU에 올릴 수 있는 모델 사이즈가 점점 커졌기 때문이다.
- 그러나 LLM(Large Language Model) 시대가 오면서 GPU 리소스는 다시 부족해지기 시작했다.
    - RTX 4090(24G) vs. Llama3 70B (150G)



https://www.etnews.com/20240319000012

GPU-efficient Machine Learning

- GPU 친화적인 코드를 짜는 것은 ML engineer와 researcher 모두에게 너무나도 중요한 역량이 되었다.
- single-GPU engineering의 시대는 끝났다!
- 그러나 학부생이 랩이나 회사 인턴이 아닌 이상 multi-GPU engineering 경험을 하긴 어렵다.



https://vast.ai/

https://vast.ai/

Large Language Model

- 초거대언어모델(LLM) 중 Meta의 Llama는 오픈소스로, 2024년 4월 말에는 llama3를 공개하였다.
- Llama3는 LLM의 여러 벤치마크에서 chatGPT보다 우수하다고 보고되었다.

LLAMA3, Meta

https://llama.meta.com/llama3/

Huggingface

- Transformer를 위한 플랫폼인 huggingface는 LLM과 관련한 ML 프레임워크를 제공한다.
- 최신 논문과 기술도 huggingface에 업로드하는 경우가 많으니 점차 익숙해지자.
- Llama3 역시 huggingface에 업로드되어 있다.

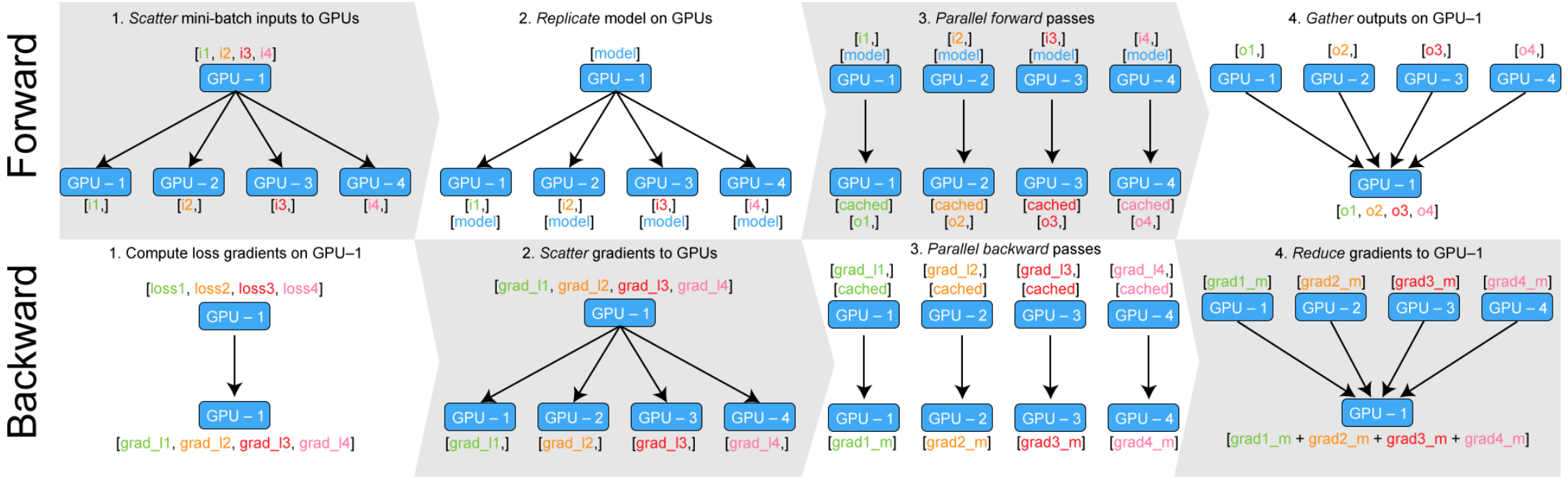
Hugging Face

https://huggingface.co/

https://huggingface.co/

Data Parallel (DP)

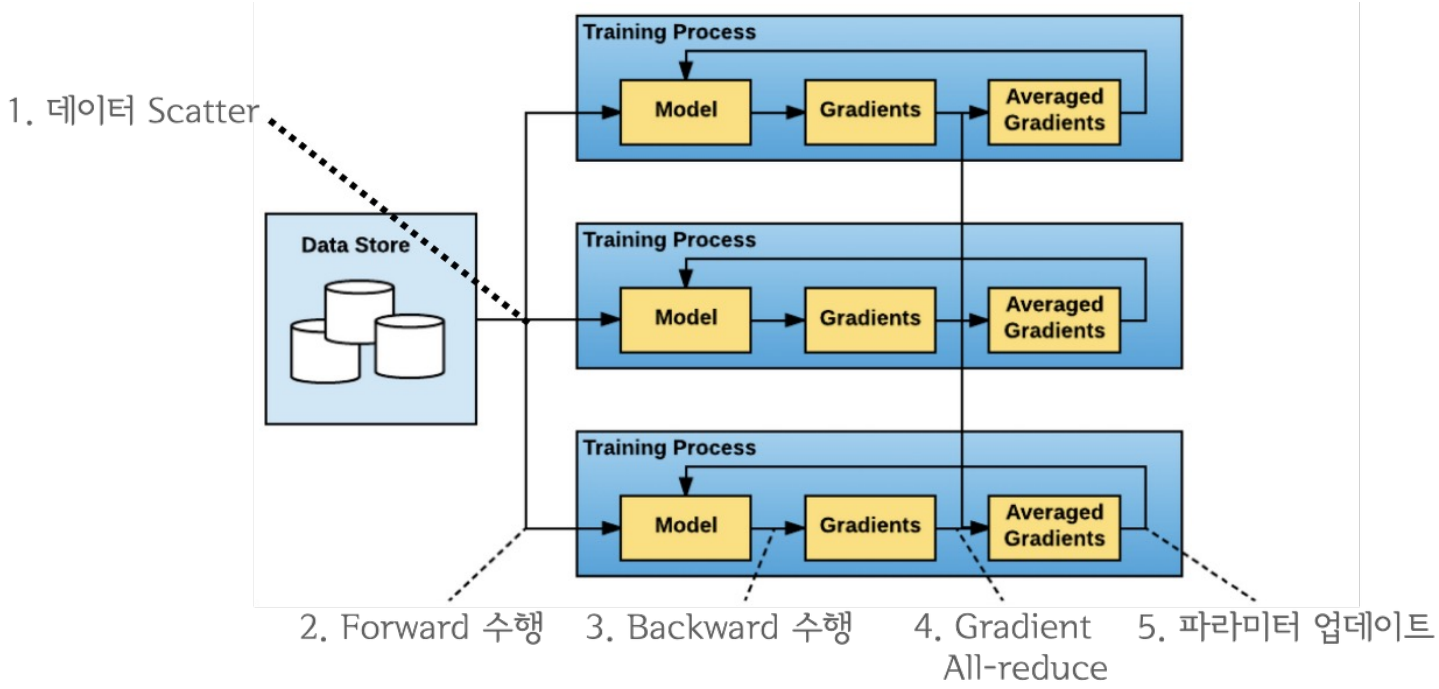
- 하나의 GPU가 master GPU가 되어서 나머지 GPU에게 minibatch와 data를 주고받는다.
- 이러한 방식은 master GPU에서 병목현상을 갖는다.
- DP는 multi-threading으로 동작한다.



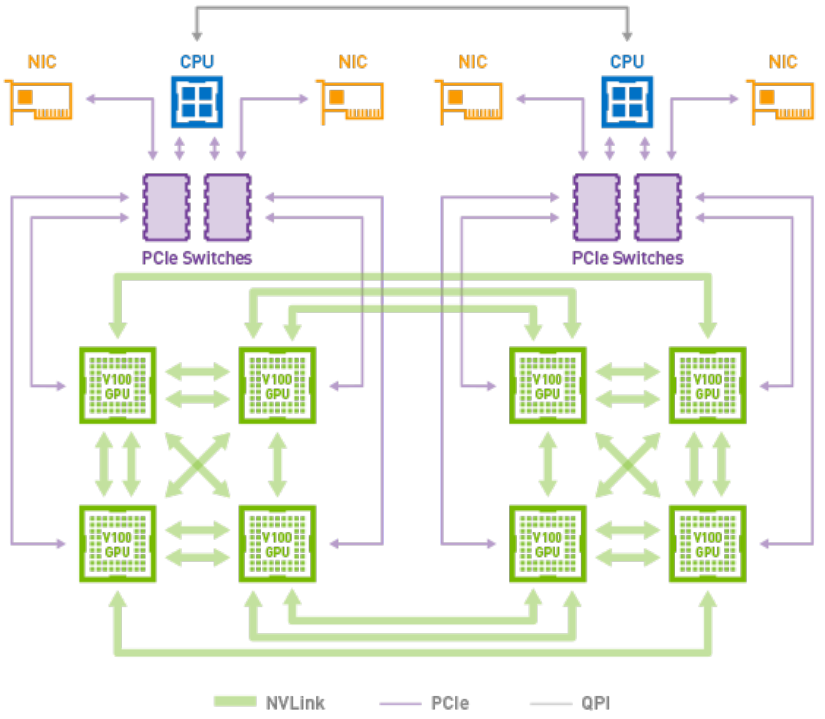
https://medium.com/huggingface/training-larger-batches-practical-tips-on-1-gpu-multi-gpu-distributed-setups-ec88c3e51255

Distributed Data Parallel (DDP)

- DDP는 각 GPU에 model, gradient, optimizer를 보관한다.
- DDP는 all-reduce를 활용하여 gradient update를 수행한다.
- 따라서 multi-node, multi-process가 가능하다.



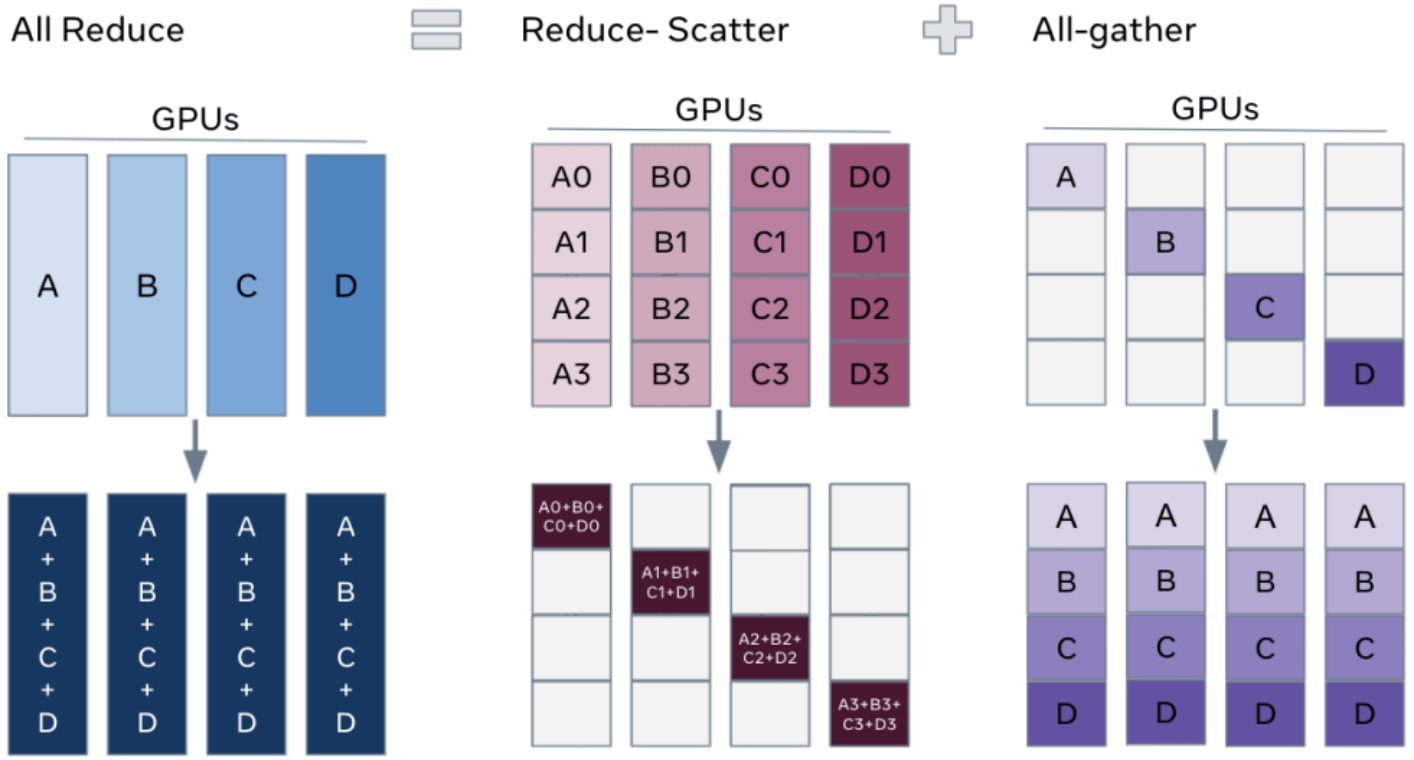
NVLink



https://blogs.nvidia.co.kr/blog/what-is-nvidia-nvlink/

Fully Sharded Data Parallel (FSDP)

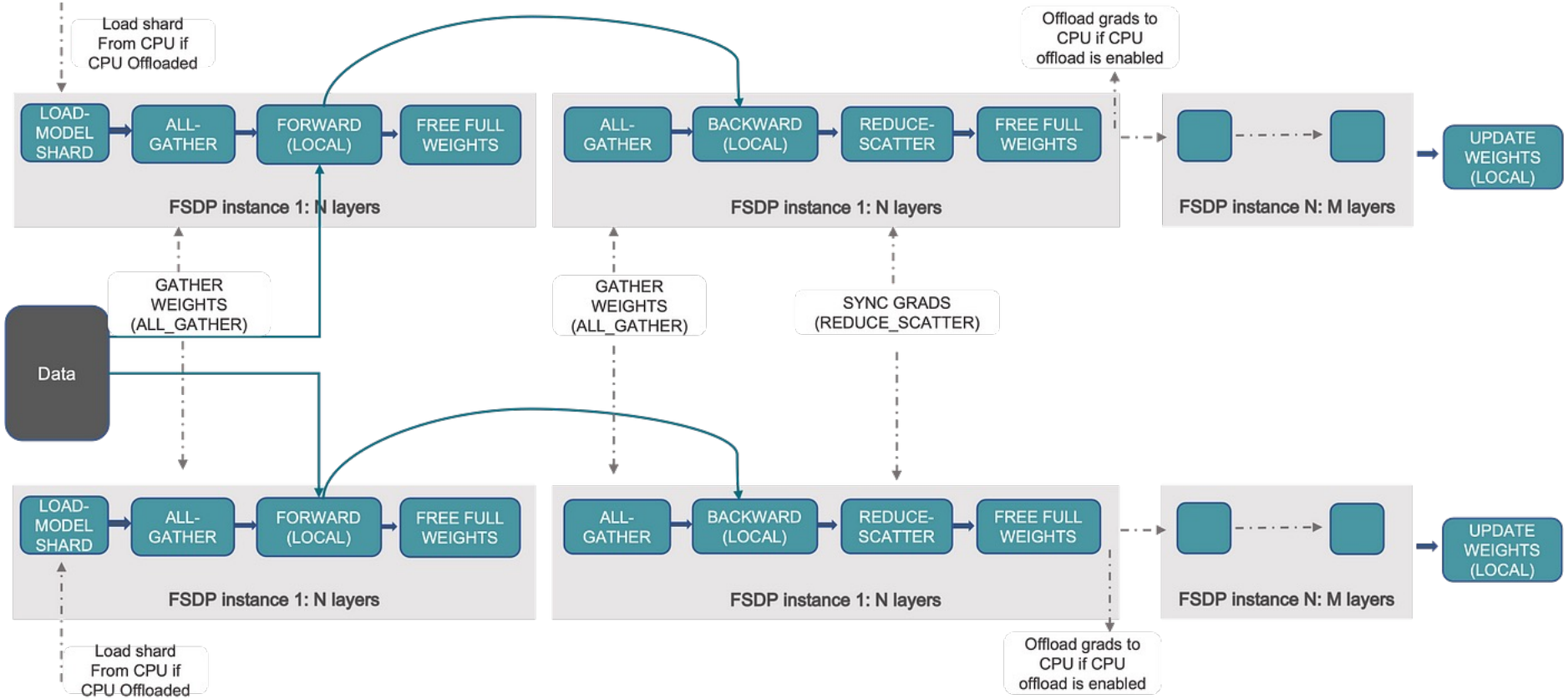
- GPU의 사이즈보다 큰 모델은 어떻게 처리할 수 있을까?
  - 각 GPU에 모델을 나눠서 저장하자.



https://engineering.fb.com/2021/07/15/open-source/fsdp/

Fully Sharded Data Parallel (FSDP)

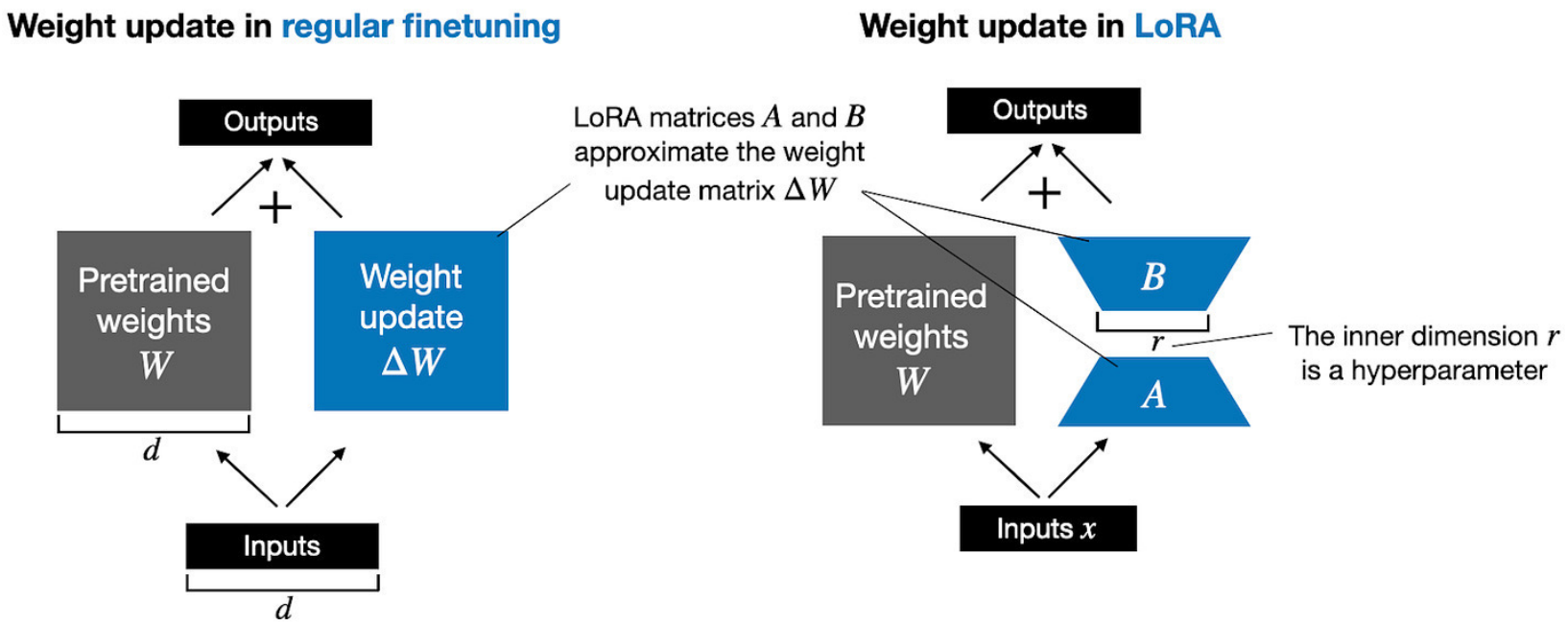
- Forwarding에서 all-gather을 하고, backwarding 할 때에도 다른 GPU의 gradient를 가져온다 (all-gather)
- 각 GPU에서는 각 layer의 gradient를 계산하고 합치는 reduce-scatter 연산을 수행한다.



https://engineering.fb.com/2021/07/15/open-source/fsdp/

Low Rank Adaptation (LoRA)

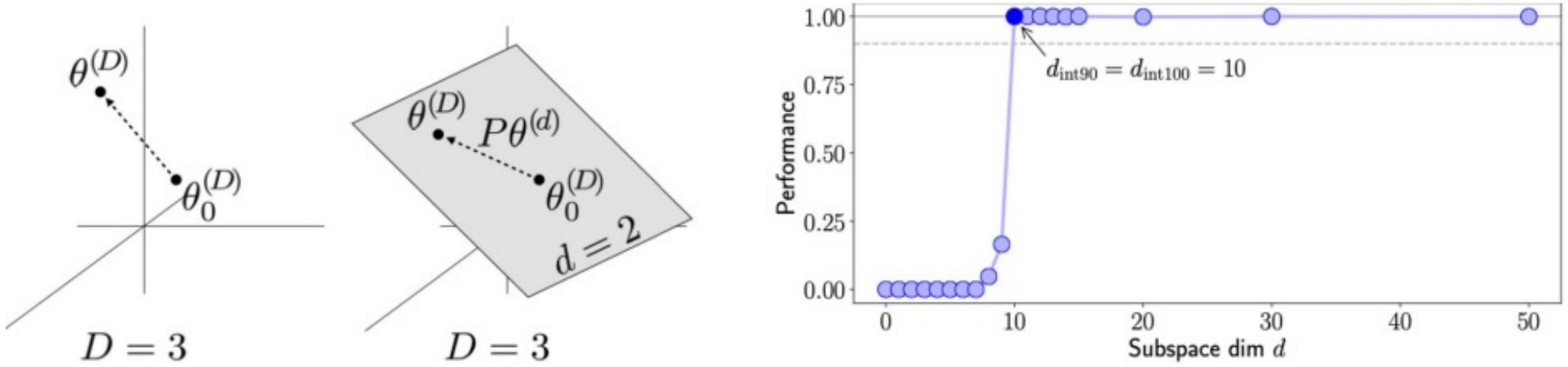
- Microsoft에서 연구한 LoRA(E. Hu et al., ICLR 22')는 memory efficient한 fine-tuning method를 제안한다.
- Pretrained-model의 가중치는 고정한 채로, 선형계층 두 개만 업데이트한다.



https://arxiv.org/abs/2106.09685

Low Rank Adaptation (LoRA)

- Microsoft에서 연구한 LoRA(E. Hu et al., ICLR 22')는 memory efficient한 fine-tuning method를 제안한다.
- Pretrained-model의 가중치는 고정한 채로, 선형계층 두 개만 업데이트한다.



https://arxiv.org/abs/2106.09685

Quantized Low Rank Adaptation (QLoRA)

- QLoRA는 모델 경량화 방법 중 하나인 양자화 방법을 LoRA에 적용한 것이다.
- (이중) 양자화에 더해 paged optimization을 통해 CPU의 RAM을 활용하면서 Out-Of-Memory를 방지한다.

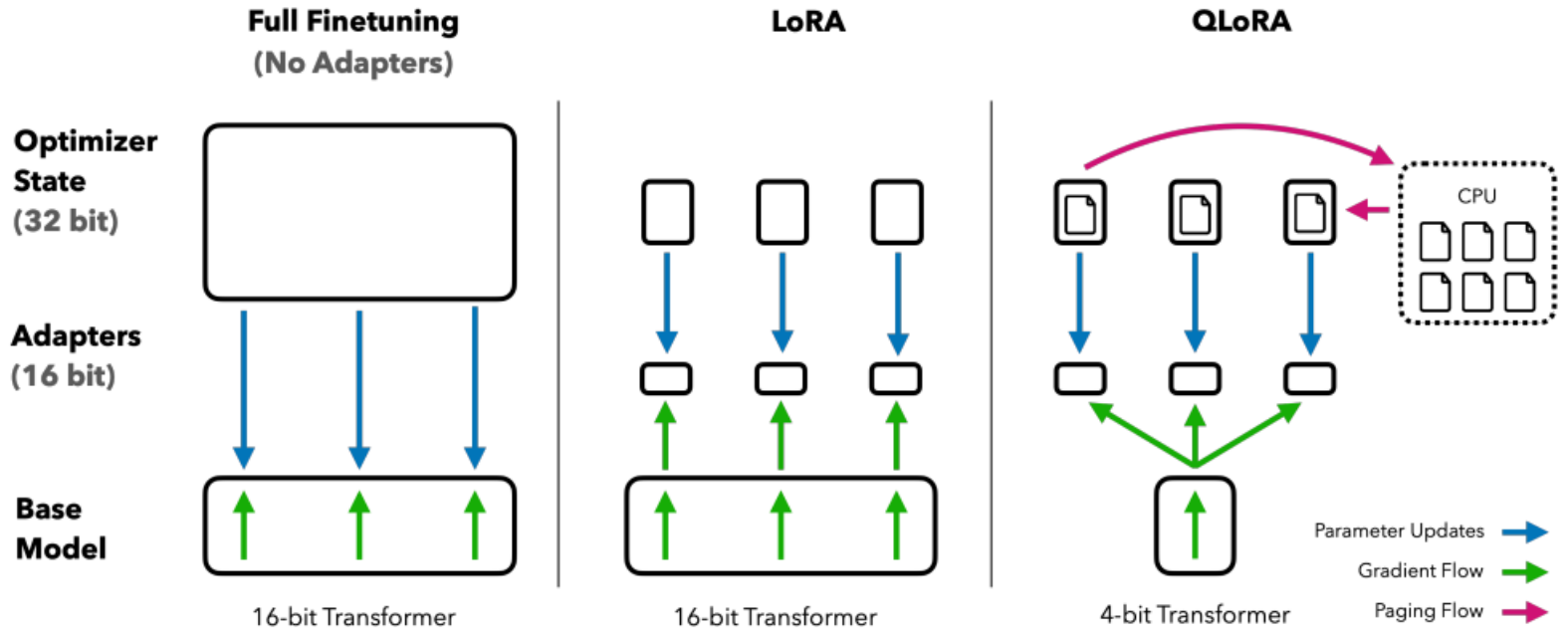


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

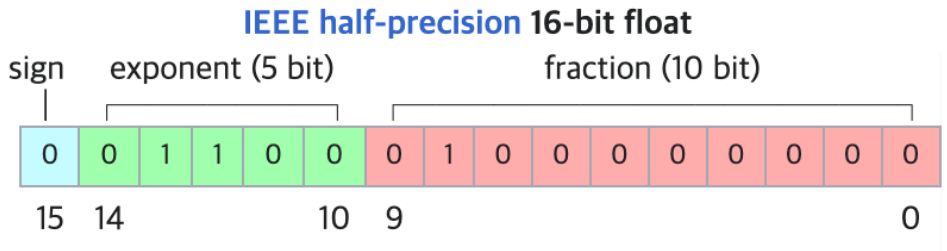
https://arxiv.org/abs/2106.09685

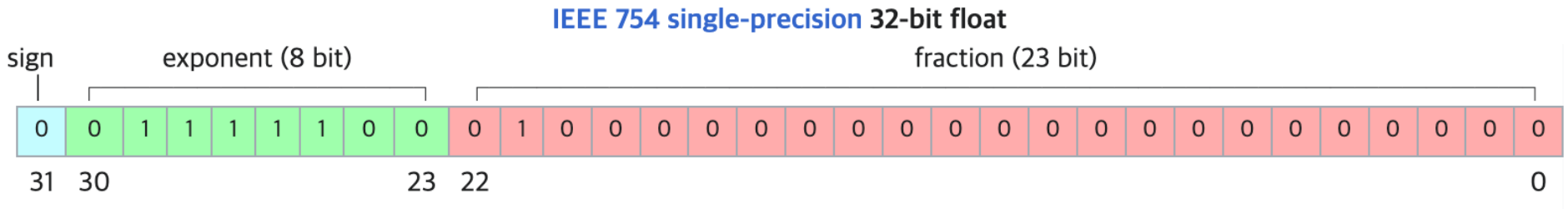
Practice

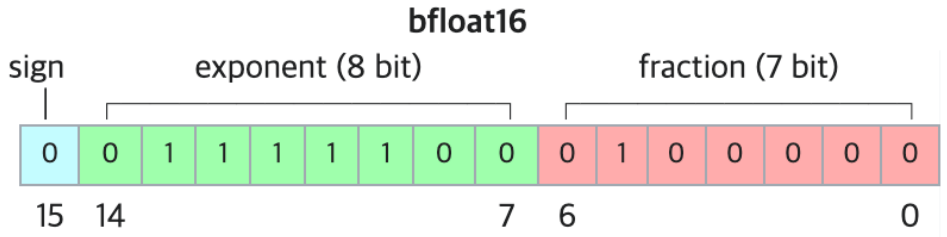
- 다양한 GPU-efficient method를 활용하여 Llama3를 finetuning해보자!

Brain Floating Point (BF16)

- 16bit에서도 지수항을 32bit와 같이 8bit로 유지한다.
- 경험적으로 training time을 50% 가량 줄여준다.



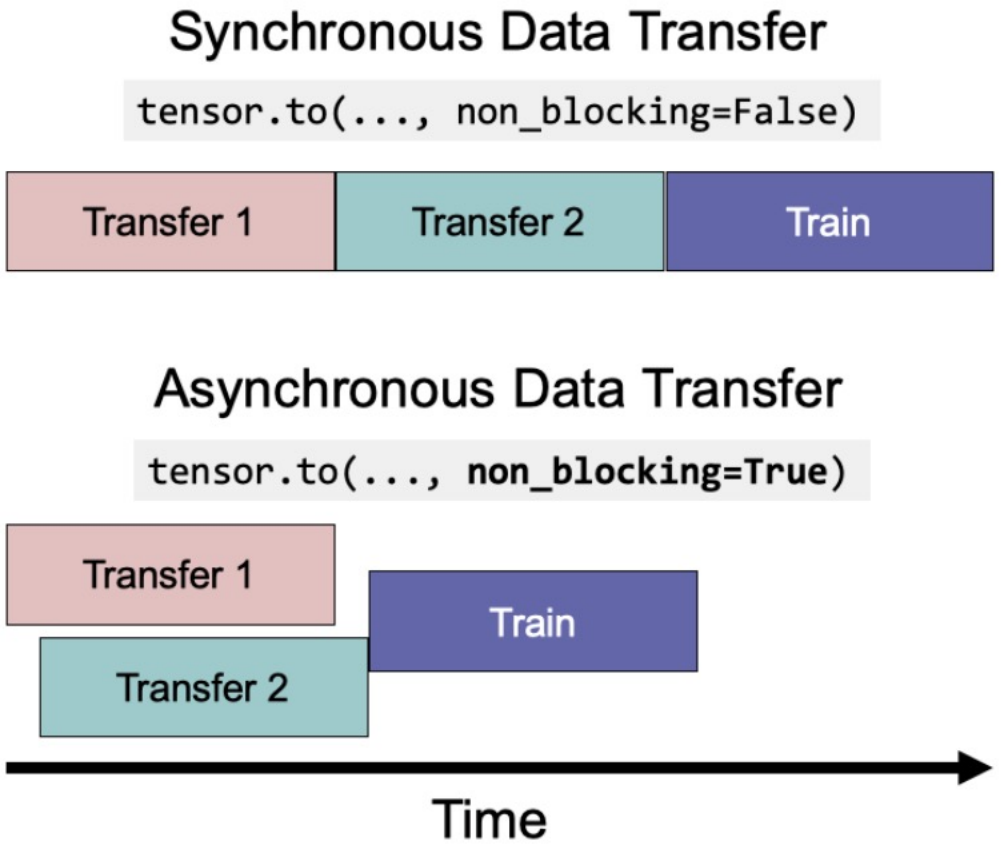




https://en.wikipedia.org/wiki/Bfloat16_floating-point_format

PyTorch non-blocking

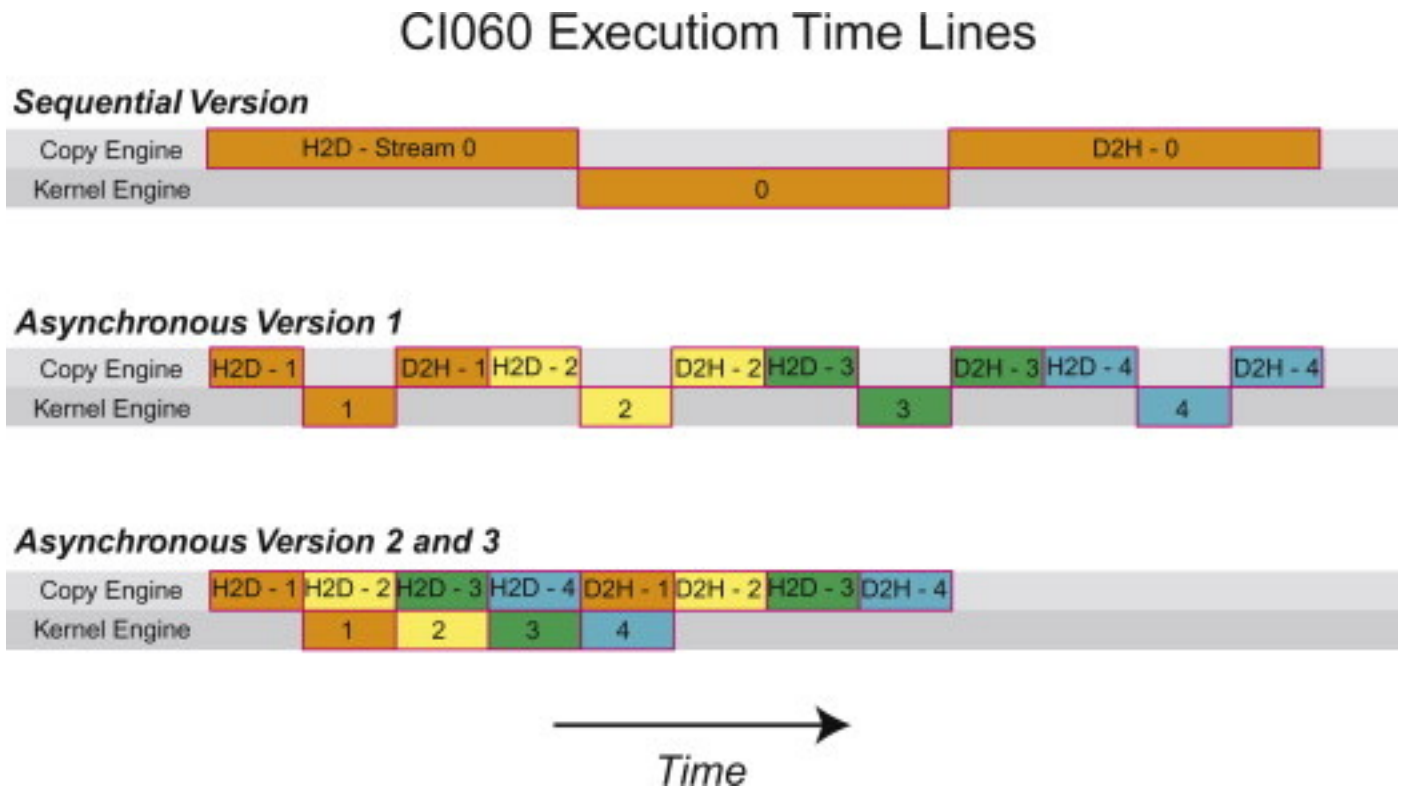
- NVidia GPU는 디바이스 연산을 위한 클락(clock)과 호스트-디바이스(CPU-GPU) 통신에 대한 클락이 따로 존재한다.
- 따라서 label 등을 CPU에서 GPU로 옮길 때 항상 non-blocking=True로 해두자.



https://miro.medium.com/max/1390/1*no-gQHz8daJbmYhCfAGNOA.png

PyTorch non-blocking

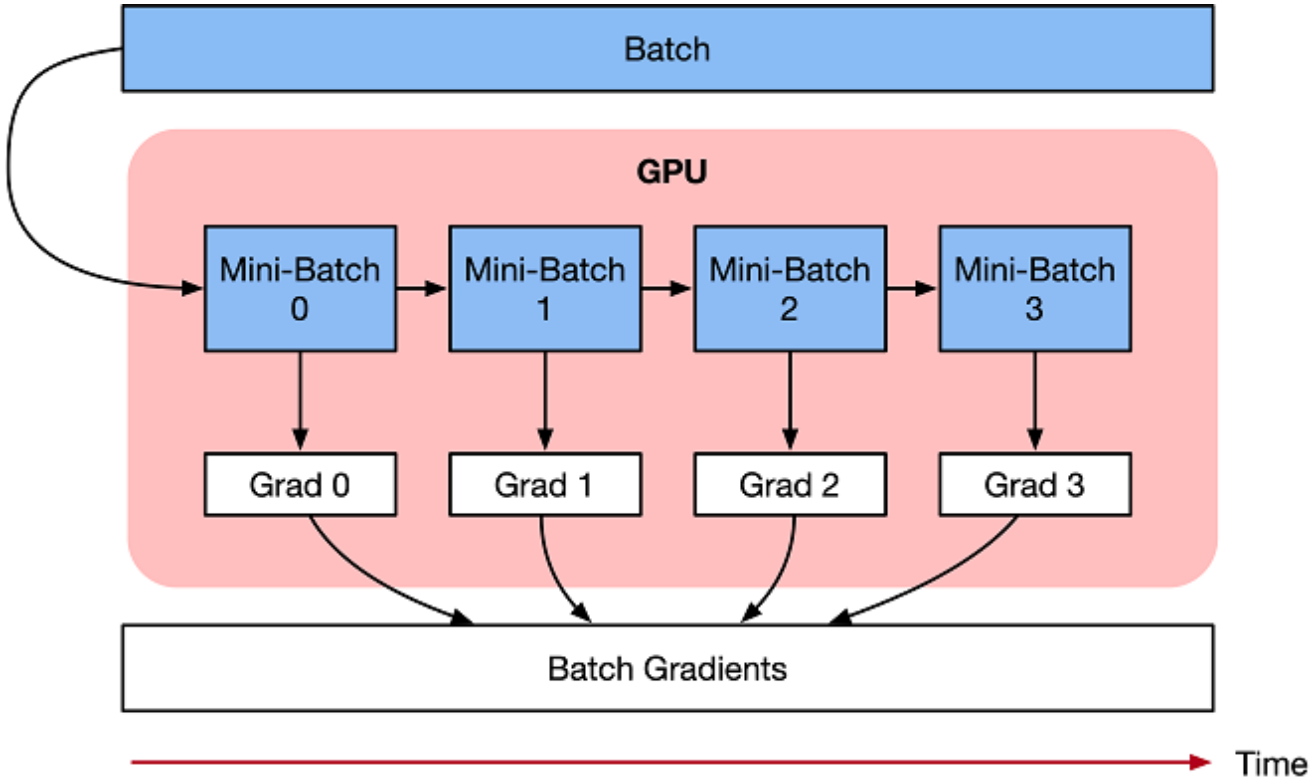
- NVidia GPU는 디바이스 연산을 위한 클락(clock)과 호스트-디바이스(CPU-GPU) 통신에 대한 클락이 따로 존재한다.
- 따라서 label 등을 CPU에서 GPU로 옮길 때 항상 non-blocking=True로 해두자.



https://www.sciencedirect.com/topics/computer-science/asynchronous-version

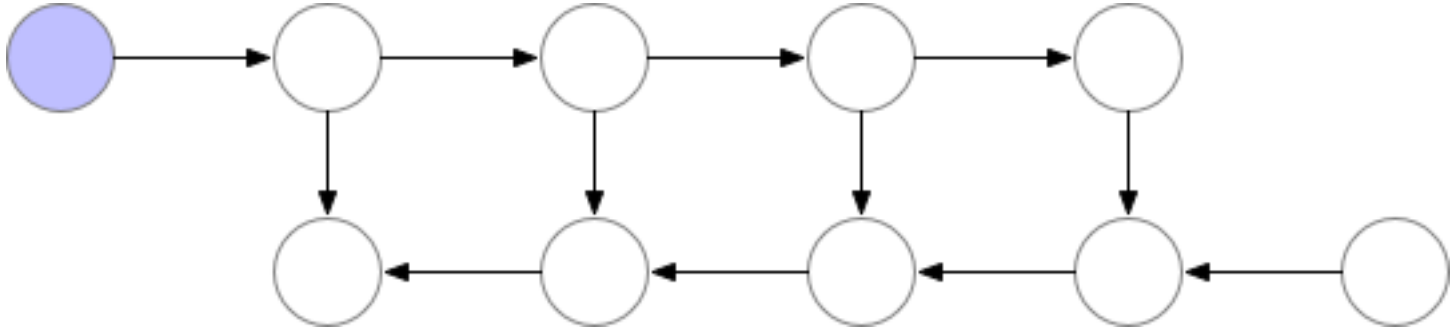
Gradient Accumulation

- OOM(Out-Of-Memory)을 회피하면서 batch size를 키우기 위한 방법이다.
- Gradient를 매 step마다 업데이트하지 않고, 모아서 한 번에 업데이트한다.
- Gradient Accumulation Law (NO SGD)

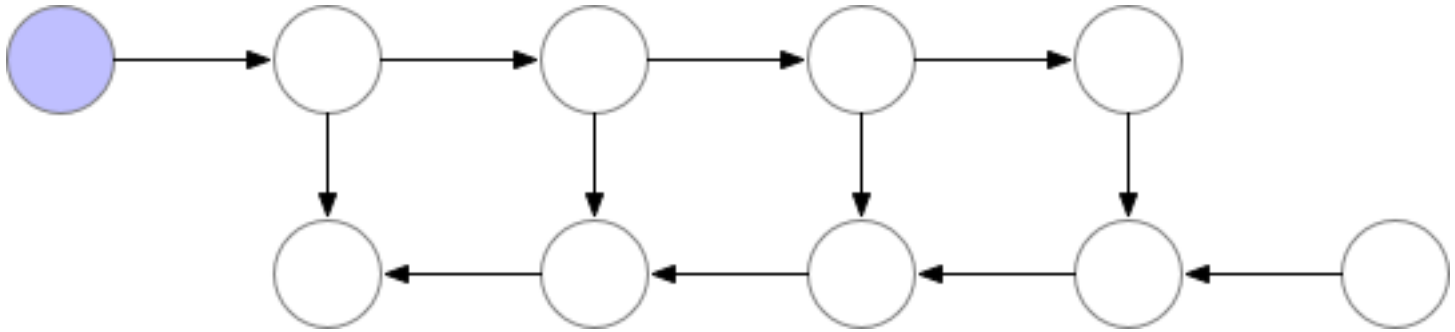


Gradient Checkpointing

- Gradient checkpointing은 역전파를 저장하여 메모리를 아낄 수 있다.



Original Propagation

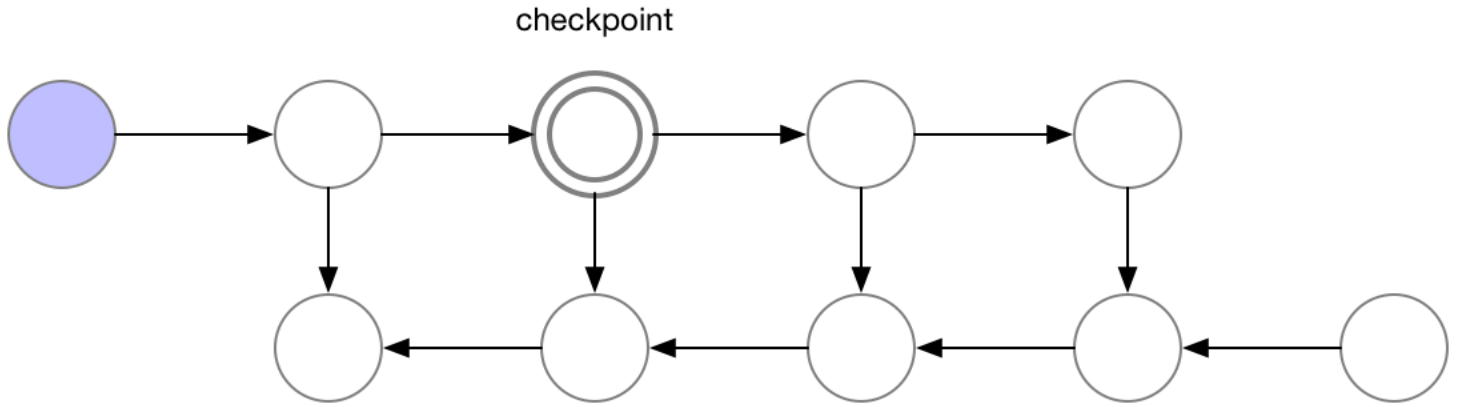


Without store gradient

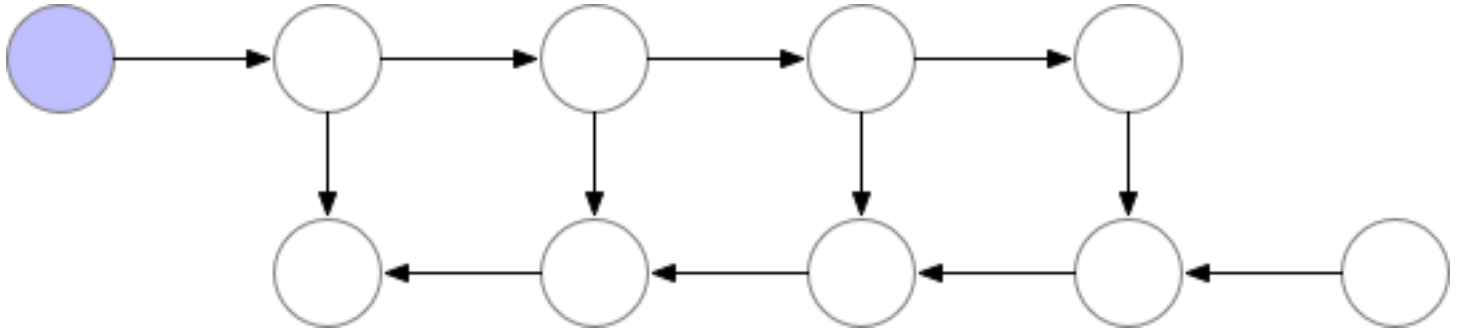
https://only-wanna.tistory.com/

Gradient Checkpointing

- Gradient checkpointing은 역전파를 저장하여 메모리를 아낄 수 있다.



With store gradient



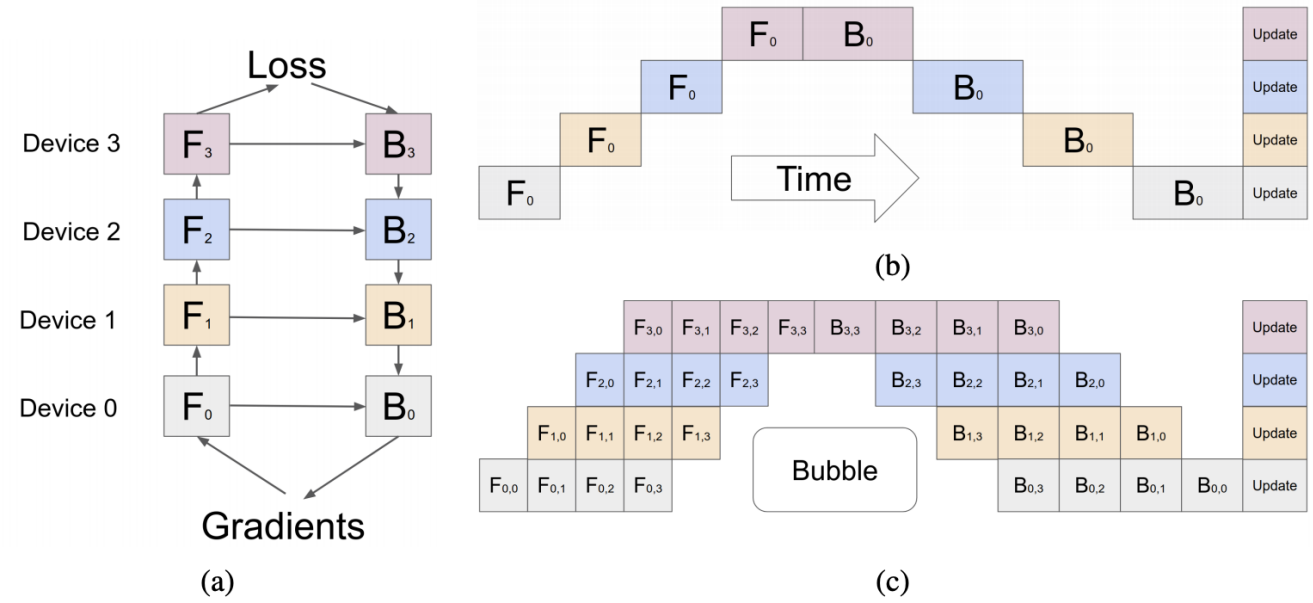
Gradient Checkpointing

https://only-wanna.tistory.com/

Gpipe

- Google이 공개한 Gpipe 라이브러리는 memory-efficient한 parallel 파이프라인을 제안한다.
- Gpipe는 minibatch보다 더 작은 micro-batch로 나누어 병렬화 성능을 극대화한다.

Figure 2: (a) An example neural network with sequential layers is partitioned across four accelerators. F_k is the composite forward computation function of the k-th cell. B_k is the back-propagation function, which depends on both B_{k+1} from the upper layer and F_k. (b) The naive model parallelism strategy leads to severe under-utilization due to the sequential dependency of the network. (c) Pipeline parallelism divides the input mini-batch into smaller micro-batches, enabling different accelerators to work on different micro-batches simultaneously. Gradients are applied synchronously at the end.

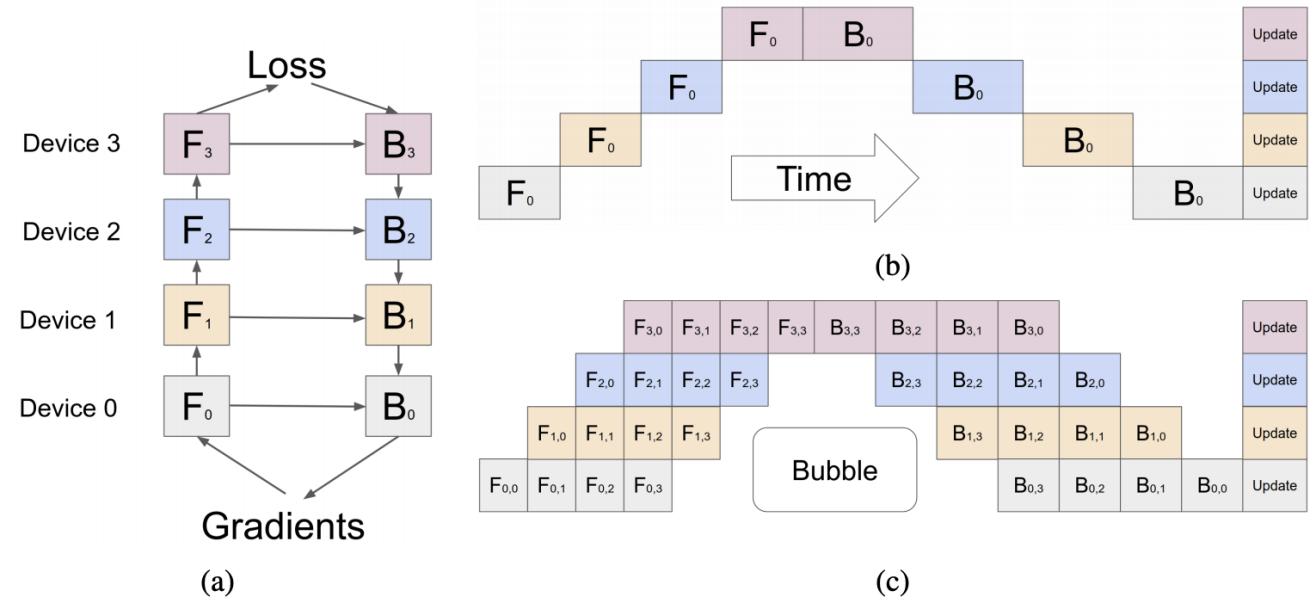


https://arxiv.org/abs/1811.06965 (Y. Huang et al., Arxiv 2018)

Gpipe

- Google이 공개한 Gpipe 라이브러리는 memory-efficient한 parallel 파이프라인을 제안한다.
- Gpipe는 minibatch보다 더 작은 micro-batch로 나누어 병렬화 성능을 극대화한다.

Figure 2: (a) An example neural network with sequential layers is partitioned across four accelerators. F_k is the composite forward computation function of the k-th cell. B_k is the back-propagation function, which depends on both B_{k+1} from the upper layer and F_k. (b) The naive model parallelism strategy leads to severe under-utilization due to the sequential dependency of the network. (c) Pipeline parallelism divides the input mini-batch into smaller micro-batches, enabling different accelerators to work on different micro-batches simultaneously. Gradients are applied synchronously at the end.



https://arxiv.org/abs/1811.06965 (Y. Huang et al., Arxiv 2018)

KUGODS

고려대학교
KOREA UNIVERSITY